Контекстно-адаптивные коды переменной длины (CAVLC)

В кодере H.264 метод CAVLC имеет дело с просканированными по зигзагу блоками 4×4, 2×4 или 2×2 квантованных коэффициентов. Таким образом, на вход CAVLC подаётся вектор из 16, 8 или 4 элементов. На выходе CAVLC получается битовая строка, соответствующая кодируемому вектору. Обычно после предсказания, преобразования и квантования блоки коэффициентов становятся весьма разреженными, т.е. содержат много нулевых коэффициентов. Для компактного представления серий нулевых коэффициентов используется схема «серия-значение», которая в случае CAVLC подразумевает отдельное кодирование ненулевых коэффициентов и отдельное кодирование серий нулей перед ними.

Завершающими элементами вектора ненулевых коэффициентов обычно являются числа ±1. В CAVLC они называются остаточными единицами (TrailingOnes) и кодируются особым образом. Остаточных единиц не может быть больше трёх. В случае, если более трёх завершающих ненулевых коэффициентов равны ±1, остаточными единицами считаются только 3 последних из них.

Битовая строка, получаемая в результате кодирования блока коэффициентов методом CAVLC, состоит из следующих синтаксических элементов:

coeff_token – код для общего количества ненулевых коэффициентов (TotalCoeff) и количества остаточных единиц (TrailingOnes).

trailing_ones_sign_flag – код для знака остаточной единицы (TrailingOne).

level_prefix – первая часть кода ненулевого коэффициента (кроме остаточных единиц).

level_suffix – вторая часть кода ненулевого коэффициента (кроме остаточных единиц).

total_zeros – код для общего количества нулей, предшествующих всем ненулевым коэффициентом (не считаются завершающие нули, стоящие после последнего ненулевого коэффициента).

run_before – код для серии нулей, предшествующей ненулевому коэффициенту.

Алгоритм CAVLC состоит из следующих шагов:

Шаг 1. Анализ входного вектора, в ходе которого вычисляются количество и значения ненулевых коэффициентов, количество нулей, стоящих перед каждым ненулевым коэффициентов и общее количество нулей.

Шаг 2. Кодирование общего количества ненулевых коэффициентов и количества остаточных единиц.

Шаг 3. Кодирование знаков остаточных единиц.

Шаг 4. Кодирование ненулевых коэффициентов.

Шаг 5. Кодирование общего количества нулей.

Шаг 6. Кодирование серий нулей, предшествующих ненулевым коэффициентам.

Подробное описание каждого шага алгоритма описано в прилагаемом документе h264.pdf на страницах 212-221.

Реализация CAVLC.

В качестве обычной реализации была взята функция кодирования алгоритмом CAVLC из проекта openh264. Ознакомиться с проектом можно на сайте https://www.openh264.org.

Варианты использования векторизации при реализации алгоритма.

1. Использование MSA для векторного кодирования элементов блока.

При данной реализации на вход будет поступать 1 массив из 16 элементов. Из этих элементов будут формироваться векторы для обработки на векторном сопроцессоре.

Рассмотрим возможность использования MSA для каждого шага алгоритма.

Шаг 1. На данном шаге не рационально использовать векторные инструкции, так как векторные инструкции не подходят для решения задач поиска элементов в массиве.

Шаг 2. Данный шаг состоит только из одной операции поиска в таблице, его реализация на MSA не рациональна.

Шаг 3. Ввиду маленького количества вычислений и простоты реализации данный шаг не нуждается в использовании векторизации.

Шаг 4. Данный шаг требует самого большого количества вычислений (из всех шагов данного алгоритма), а возможность параллельного кодирования коэффициентов позволяет реализовать его на MSA. Однако, реализация MSA затрудняется по следующим двум причинам. а) существует переменная, которая зависит от кодирования предыдущих коэффициентов; б) Наличие 3 if-else операторов, реализация которых на MSA требует использования дополнительных MSA инструкций. Вычисление значения переменной до цикла позволяет решить а), а использование дополнительных MSA инструкций позволяет решить б).

Шаг 5. Данный шаг состоит только из одной операции поиска в таблице, его реализация на MSA не рациональна.

Шаг 6. Каждый шаг кодирования серии нулей требует информации о количестве оставшихся нулей, а само кодирование происходит путем поиска в таблице. Выбор кода зависит от количества оставшихся нулей и количества нулей в серии, что делает нерациональным кодирование серий нулей инструкциями MSA.

Результаты реализации.

В зависимости от вида входного вектора процент использования MSA инструкций может меняться. Рассмотрим вида входных данных:

- Почти все нулевые коэффициенты
- Равное количество нулевых и ненулевых коэффициентов
- Все ненулевые коэффициенты

Количество итераций составляло 1000.

Почти все нулевые коэффициенты.

Входная последовательность:

0, 3, 0, 0, 0, 1, 0, 0, 0, -1, 0, -1, 0, 0, 0, 0

Выполненные инструкции:

Название команд	Количество вызовов
ADDV.df	8000
ADDVI.df	1000
AND.V	11000
CEQI.df	2000
CLTI_D.df	2000
CLT_S.df	1000
LD.df	52000
LDI.df	7000
MOVE.V	1000
NOR.V	3000
OR.V	1000
SLL.df	3000
SLLI.df	2000
SRA.df	1000
SRAI.df	1000
ST.df	23000
SUBV.df	10000
SUBVI.df	1000
XOR.V	1000
OPC_ARITH	99998
OPC_ARITH_IMM	90000
OPC_COND_MOVE	14000
OPC_LD	344001
OPC_LOGIC	63000
OPC_LOGIC_IMM	8000
OPC_SHIFT	2000
OPC_SHIFT_IMM	78000
OPC_SLT_IMM	8000
OPC_ST	134001
OPC_ST_COND	1

Количество операций MSA: 131000

Количество обычных операций: 841001

Процент команд MSA: 13.5%

Равное количество нулевых и ненулевых коэффициентов.

Входная последовательность:

0, -5, 4, 0, 0, -3, 3, 0, 0, 2, 0, -1, 1, 1, 0, 0

Выполненные инструкции:

Название команд	Количество вызовов
ADDV.df	16000
ADDVI.df	2000
AND.V	22000
CEQI.df	4000
CLTI_D.df	4000
CLT_S.df	2000
LD.df	104000

LDI.df	14000
MOVE.V	2000
NOR.V	6000
OR.V	2000
SLL.df	6000
SLLI.df	4000
SRA.df	2000
SRAI.df	2000
ST.df	46000
SUBV.df	20000
SUBVI.df	2000
XOR.V	2000
OPC_ARITH	189998
OPC_ARITH_IMM	142000
OPC_COND_MOVE	26000
OPC_LD	583001
OPC_LOGIC	100000
OPC_LOGIC_IMM	15000
OPC_SHIFT	6000
OPC_SHIFT_IMM	131000
OPC_SLT_IMM	18000
OPC_ST	207001
OPC_ST_COND	1

Количество операций MSA: 262000

Количество обычных операций: 1418001

Процент команд MSA: 15.6%

Все ненулевые коэффициенты.

Входная последовательность:

12, -10, -8, 6, 6, -5, 4, 4, 3, -3, -3, 2, 2, -1, -1, 1

Выполненные инструкции:

Название команд	Количество вызовов
ADDV.df	32000
ADDVI.df	4000
AND.V	44000
CEQI.df	8000
CLTI_D.df	8000
CLT_S.df	4000
LD.df	208000
LDI.df	28000
MOVE.V	4000
NOR.V	12000
OR.V	4000
SLL.df	12000
SLLI.df	8000
SRA.df	4000
SRAI.df	4000
ST.df	92000
SUBV.df	40000

SUBVI.df	4000
XOR.V	4000
OPC_ARITH	234999
OPC_ARITH_IMM	182001
OPC_COND_MOVE	50000
OPC_LD	764007
OPC_LOGIC	102000
OPC_LOGIC_IMM	32000
OPC_SHIFT	14000
OPC_SHIFT_IMM	175001
OPC_SLT_IMM	38000
OPC_ST	261001
OPC_ST_COND	1

Количество операций MSA: 524000
Количество обычных операций: 1853010
Процент команд MSA: 28.3%

Из таблиц видно, что чем больше ненулевых коэффициентов, тем больше процент операций MSA, что обусловлено применением векторизации для кодирования ненулевых коэффициентов.

2. Использование MSA для векторного кодирования блоков.

При данной реализации на вход алгоритма будут поступать 4 массива по 16 элементов в каждом. Из этих массивов будут составлены 16 векторов по 4 элемента в каждом. Элементы вектора – коэффициенты из массивов, имеющие одинаковый индекс. Поэтому инструкции MSA будут применяться ко всем коэффициентам в массиве с одинаковыми индексами.

Рассмотрим возможность использования MSA для каждого шага алгоритма.

Шаг 1. На данном этапе происходит анализ входных массивов и формируются векторы с ненулевыми коэффициентами, вектор-количество ненулевых коэффициентов в каждом массиве и вектора с количеством нулевых коэффициентов, стоящих перед ненулевыми коэффициентами. Основной задачей данного шага является формирование из неупорядоченных данных MSA векторов для их последующей обработки.

Шаг 2. Ввиду использования таблицы, кодирование общего количества ненулевых коэффициентов и остаточных единиц не нуждается в использовании инструкций MSA.

Шаг 3. Так как для кодирования остаточных единиц используются битовые операции, данные в векторах остаточных единиц не зависят друг от друга, то целесообразно использование инструкций MSA.

Шаг 4. Данный шаг требует большого количества вычислений и хорошо реализуется на MSA. Однако ввиду разного количества ненулевых коэффициентов в каждом из массивов, появляются избыточные операции. Так же на данном шаге присутствуют if-else, что так же вводит дополнительные инструкции MSA.

Шаг 5. Данный шаг состоит только из одной операции поиска в таблице, его реализация на MSA не рациональна.

Шаг 6. В данном шаге используется 2 операции поиска и 1 арифметическая операции, а реализация данного шага на MSA использует дополнительные инструкции, поэтому такая реализация не рациональна.

Результаты реализации.

В зависимости от вида входных данных процент использования MSA инструкций может меняться. Рассмотрим вида входных данных:

- Все векторы имеют много нулевых коэффициентов
- Все векторы имеют мало ненулевых коэффициентов
- Векторы имеют большую дисперсию количества нулевых и ненулевых коэффициентов

Количество итераций составляло 1000.

Все векторы имеют много нулевых коэффициентов

Название команд	Количество вызовов
ADDV.df	50006
ADDVI.df	4000
AND.V	100012
CEQ.df	4000
CEQI.df	34006
CLEI_S.df	1000
CLTI_D.df	17000
CLT_S.df	12000
COPY_S.df	2000
FILL.df	8000
INSERT.df	66024
LD.df	455060
LDI.df	126030
MOVE.V	20000
NOR.V	53012
OR.V	37006
SLL.df	20000
SLLI.df	8000
SRA.df	4000
SRAI.df	4000
SRLI.df	3000
ST.df	262054
SUBV.df	76006
SUBVI.df	8000
XOR.V	4000
OPC_ARITH	682144
OPC_ARITH_IMM	346037
OPC_LD	1798434
OPC_LOGIC	182036
OPC_LOGIC_IMM	1000
OPC_SHIFT_IMM	517108

OPC_ST	330056

Количество операций MSA: 1378216
Количество обычных операций: 3856815
Процент команд MSA: 73.7%

Все векторы имеют мало ненулевых коэффициентов

Название команд	Количество вызовов
ADDV.df	122027
ADDVI.df	13000
AND.V	217084
CEQ.df	13000
CEQI.df	61039
CLEI_S.df	1000
CLTI_D.df	53000
CLT_S.df	39000
COPY_S.df	2000
FILL.df	26000
INSERT.df	66108
LD.df	995354
LDI.df	207141
MOVE.V	56006
NOR.V	71078
OR.V	64045
SLL.df	56006
SLLI.df	26000
SRA.df	13000
SRAI.df	13000
SRLI.df	3006
ST.df	469273
SUBV.df	202033
SUBVI.df	26000
XOR.V	13000
OPC_ARITH	612667
OPC_ARITH_IMM	321176
OPC_LD	1693978
OPC_LOGIC	158162
OPC_LOGIC_IMM	1000
OPC_SHIFT_IMM	471505
OPC_ST	311259

Количество операций MSA: 2828200
Количество обычных операций: 3569747
Процент команд MSA: 55.8%

Векторы имеют большую дисперсию количества нулевых и ненулевых коэффициентов

Название команд	Количество вызовов
ADDV.df	122504

ADDVI.df	13062
AND.V	217822
CEQ.df	13062
CEQI.df	61194
CLEI_S.df	1000
CLTI_D.df	53248
CLT_S.df	39186
COPY_S.df	2000
FILL.df	26124
INSERT.df	66032
LD.df	998800
LDI.df	207604
MOVE.V	56248
NOR.V	71140
OR.V	64194
SLL.df	56248
SLLI.df	26124
SRA.df	13062
SRAI.df	13062
SRLI.df	3000
ST.df	470504
SUBV.df	202876
SUBVI.df	26124
XOR.V	13062
OPC_ARITH	665254
OPC_ARITH_IMM	350177
OPC_LD	1793890
OPC_LOGIC	174050
OPC_LOGIC_IMM	1000
OPC_SHIFT_IMM	508206
OPC_ST	336210

Количество операций MSA: 2837282
Количество обычных операций: 3828787
Процент команд MSA: 57.4%

Достоинство данной реализации состоит в том, что она позволяет обрабатывать одновременно 4 блока, однако, если блоки разного типа (сильно различное количество нулевых и ненулевых коэффициентов), то появляются избыточные команды.